

Pages 1 - 60

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

BEFORE THE HONORABLE WILLIAM H. ALSUP

ORACLE AMERICA, INC.,)	
)	
Plaintiff,)	
)	
vs.)	NO. C 10-03561 WHA
)	
GOOGLE INC.,)	
)	San Francisco, California
Defendant.)	Wednesday
)	April 6, 2011
)	1:41 p.m.

TRANSCRIPT OF PROCEEDINGS

APPEARANCES:

For Plaintiff:	Morrison & Foerster 755 Page Mill Road Palo Alto, CA 94304-1018 (650) 813-5600 (650) 494-0792 (fax)
BY:	Michael A. Jacobs Marc David Peters Ruchika Agrawal Roman Swoopes
For Plaintiff:	Oracle 500 Oracle Parkway M/S 50P7 Redwood Shores, CA 94065 (650) 506-9432 (650) 506-7114 (fax)
BY:	Andrew C. Temkin

(Appearances continued on next page)

Reported By: Lydia Zinn, CSR #9223, RPR
Official Reporter - U.S. District Court

1 APPEARANCES (CONT'D)

2 **For Defendant:** King & Spalding LLP
1185 Avenue of the Americas
3 New York, NY 10036-4003
(212) 556-2100

4 **BY: Scott T. Weingaertner**
Mark H. Francis

5 King & Spalding
6 1180 Peachtree Street, N.E.
Atlanta, GA 30309-3521
7 (404) 572-4826
(404) 572-5134 (fax)

8 **BY: Bruce W. Baber**

9 King & Spalding
100 N Tryon Street, Suite 2900
10 Charlotte, NC 28202
(704) 503-2630
11 704) 503-2622 (fax)

12 **BY: Steven T. Snyder**

13 King & Spalding
101 Second Street, Suite 2300
San Francisco, CA 94105
14 (415) 318-1220
(415) 318-1300 (fax)

15 **BY: Donald Frederick Zimmer, JR.**

16 **For Defendant:** Google
1600 Amphitheatre Parkway
17 Mountain View, CA 94043
(650) 253-0000
18 (650) 618-1806 (fax)

19 **BY: Renny Hwang**

20 **Also Present: John Mitchell**

1 **THE COURT:** Good afternoon, everyone. Welcome back.
2 Please be seated.

3 Okay. We're here in the case of *Oracle versus*
4 *Google*. Number 10-3561. Please make your appearances.

5 **MR. JACOBS:** Michael Jacobs, from
6 Morrison & Foerster, your Honor, for Oracle.

7 With me at counsel table is Ruchika Agrawal, who,
8 consistent with your Honor's guidance, will have a speaking
9 role in the tutorial today. And then next to her is
10 Roman Swoopes, who's assisting on the tutorial;
11 Professor John Mitchell, of the Computer Science Department at
12 Stanford University, our retained expert in the case;
13 Andrew Temkin, corporate counsel at Oracle; and Marc Peters, my
14 partner, who'll be handling the discovery issues this morning.

15 **THE COURT:** All right. And on the other side.

16 **MR. WEINGAERTNER:** Good afternoon, your Honor.
17 Scott Weingaertner, of King & Spalding, for Google.

18 Here with me at counsel table are Bruce Baber;
19 Renny Hwang, of Google; Steve Snyder, also of King & Spalding;
20 Fritz Zimmer, King & Spalding; and Mark Francis, also of
21 King & Spalding, who will be assisting with the presentation
22 today.

23 **THE COURT:** All right. Here's what we will do. I
24 know I noticed the discovery hearing for this. So we'll take
25 that up last, if time permits. If it doesn't permit, then I

1 may have to refer all discovery in this case to a Magistrate
2 Judge, because I'm in a trial that takes from 5:00 a.m. until
3 1:00 every day, and I run out of time.

4 I'm going to do your claim construction.

5 Yes, sir.

6 **MR. BABER:** Your Honor, I was just going to say --
7 and Mr. Peters can jump in, but we did have a plus-four hours
8 meet-and-confer. We have reached agreement on a number of
9 things. I believe it's going to be a question of reading into
10 the record the agreements we reached.

11 **THE COURT:** How long will that take?

12 **MR. BABER:** Probably less than five minutes.

13 **THE COURT:** All right. Well, then, does that end it,
14 or is there more for me to rule on?

15 **MR. BABER:** At this point, your Honor, I don't know
16 if there's any more for you to rule on.

17 **THE COURT:** All right. Then let's put it on the
18 record, and then we'll go to your tutorial. So that's good.
19 That's good. Thank you for doing that.

20 Okay. Mr. Jacobs said he would be arguing the
21 discovery.

22 Is there more -- is that true? -- to put on the
23 record?

24 **MR. PETERS:** It will be me, your Honor. There won't
25 be anything to argue if we just put this on the record.

1 **THE COURT:** Great. You two come forward. Here's the
2 way you do it. You state what the agreement is. Pause after
3 each sentence. And then the other side says, "Agreed," or "Not
4 agreed," as the case may be.

5 Go ahead.

6 **MR. BABER:** Okay. Your Honor, we did discuss a lot
7 of issues. We reached a agreement on a number of them. One by
8 one. First are some process steps.

9 We agreed that if either party has raised an issue by
10 e-mail or by letter to the other side, and hasn't heard back in
11 a reasonable period of time for a response, that party will
12 follow up, and simply remind the other party that the request
13 is outstanding. In response to the follow-up, the other side
14 will, by the close of next business day, respond with either
15 some communication saying, "We're not going to do what you
16 asked us to," or "We are looking into it, and we expect to
17 provide a substantive response by a date certain." Just a
18 process. Sort of make sure that communications keep flowing
19 appropriately.

20 **MR. PETERS:** We agree to that procedure, your Honor.

21 **THE COURT:** And before you go any further, can I make
22 a comment?

23 **MR. BABER:** Yes, sir.

24 **THE COURT:** I have a pretty strong rule -- not rule,
25 but encouragement -- to get young people to argue in court.

1 On the other hand, I have found that if you delegate
2 down to young people the discovery disputes, which often
3 happens, they don't get resolved, and that they actually get
4 resolved better if you give them the higher up the chain you
5 go, because the lawyers higher up the chain have the competence
6 to give something up. So I encourage you to either have your
7 number-one or number-two person on the case to do that second
8 letter.

9 In other words, if you're going to write back and
10 say, "We're not going to do anything more on that," or "Here's
11 all we're going to do, and that's the end of it," I encourage
12 it to be number one or number two, and not a lower-down person.

13 **MR. BABER:** Your Honor, I think you've read our
14 collective minds.

15 Second point is that every ten days, from now until
16 the time of trial, we will have a telephone conference in the
17 case, in which the senior-level lawyers on both sides will
18 participate; either Mr. Weingaertner, myself, or Mr. Perry for
19 Google; Mr. Jacobs, Mr. -- Dr. Peters, or Mr. Holtzman for
20 Oracle, depending on the issues. We will have an agenda two
21 days in advance of those calls, so any discovery or other
22 issues that are out there that either side wants to talk about,
23 we will have a set time for the senior folks on the case to
24 talk about them, and try to resolve them.

25 And we have undertaken to send to Oracle, by the end

1 of this week, a proposed, specific schedule for the every ten
2 days from here until the time of trial.

3 **MR. PETERS:** That's right, your Honor.

4 **THE COURT:** Great. Wonderful.

5 **MR. BABER:** Okay. Point number three, your Honor, is
6 on supplementation of interrogatory answers, we have agreed
7 that both parties will supplement interrogatory answers by
8 Friday, April 22nd, which is two weeks from this Friday. And
9 then we also have some additional supplementation from Google.
10 I'll go into the details in a second. The broad parameter is
11 April 22nd.

12 **MR. PETERS:** That's right.

13 **THE COURT:** Stop on that. That's good.

14 I want just say to you both I won't make a ruling on
15 this now, but if I was in your position, I would have an
16 exceedingly good answer, because I may encourage somebody to
17 bring a motion to knock out a defense or knock out a claim for
18 insufficient basis. So if you want to protect yourself against
19 that, you should err on the side of more disclosure and more
20 answer, and not hide the ball. Okay?

21 **MR. BABER:** Yes, your Honor.

22 **THE COURT:** That's just a word to the wise.

23 Okay. What's your next --

24 **MR. BABER:** Next point, your Honor, is a little more
25 specific. On the supplementation of interrogatory answers,

1 Oracle has agreed that it will supplement its responses to
2 Google's Interrogatories 1 through 10 by April the 22nd, and
3 will include in those supplemental responses any additional
4 information that supports its contentions or positions, and/or
5 to take out any items that it no longer relies on to the extent
6 its contentions have changed, and things that may have been
7 identified as potentially infringing early on now may drop out
8 of the equation, but that will be Oracle's supplementation.

9 Google's supplementation by April 22nd is that we
10 will supplement our responses to Interrogatories 4. We will
11 supplement our response to Number 5, if response to Oracle's
12 recently supplemented infringement contentions. We will also
13 supplement Number 6, 7, 8, 10 through 16 in the first instance.

14 And then on April 29th, after we've received their
15 supplements, we will supplement again as to our answers to
16 Number 9 and 10, which are copyright-related ones that deal
17 with specific files. So we need to see what they come back at
18 us with before we can respond.

19 **MR. PETERS:** That's our agreement, your Honor.

20 **THE COURT:** Wonderful. Next.

21 **MR. BABER:** Your Honor, we also discussed Google's
22 response to Oracle's Interrogatory Number 3, which relates to
23 noninfringement. We have agreed that by next Monday, we will
24 send to Oracle's counsel a letter summarizing what we think we
25 heard from them in the meet-and-confer, and clarify what more

1 it is they want us or would ask us to put in that interrogatory
2 answer. They will respond to that, and confirm that either
3 that is or is not the case. And we will then see what's on the
4 table, and we will supplement it to the extent we believe it
5 needs supplementation. And we'll do it within 14 days after
6 that.

7 Once we have supplemented again on noninfringement,
8 Oracle will then come back to us and tell us where they
9 disagree with any new information we've shared with them. In
10 other words, if we say, "It doesn't -- it's not covered by a
11 claim for this reason," and then they'll come back and, if we
12 ask them to, they'll tell us, "Well, we say it is, and here's
13 why." So again, back and forth on a 14-day framework.

14 **MR. PETERS:** That's right, your Honor.

15 **THE COURT:** Okay.

16 **MR. BABER:** With respect to supplemental
17 interrogatory answers generally, the parties have agreed in
18 concept aspirationally to the concept of parity. That is, both
19 parties will make their best efforts in good faith to provide a
20 comparable level of particularity, granularity, detail --
21 whatever you want to call it -- that is being provided by the
22 other side.

23 **THE COURT:** Isn't that what you agreed to last time?

24 **MR. BABER:** I beg your pardon, your Honor?

25 **THE COURT:** That's what both sides agreed to last

1 time.

2 **MR. BABER:** Well, we thought so as well.

3 **THE COURT:** It fell apart.

4 **MR. BABER:** I'm not sure it fell apart. We may be
5 moving to a different level of granularity at this point. And
6 so we will try to be sure we are each providing apples and
7 apples. And that's true whether it's a question of fact, a
8 contention, or, where appropriate, the application of fact to
9 law.

10 **MR. PETERS:** Consistent with your Honor's earlier
11 indication about getting as much information as possible
12 through interrogatory responses, that is our agreement.

13 **THE COURT:** Great.

14 **MR. BABER:** Next issue, your Honor, has to do,
15 instead of interrogatories, with the plaintiff's infringement
16 contentions under the patent local rules.

17 Oracle has declined Google's offer to allow Oracle to
18 supplement its infringement contentions again. Oracle has
19 chosen to rely on its infringement contentions as currently
20 framed.

21 Google's position that they are inadequate, in
22 particular on the issue of proof of direct infringement; but
23 the parties have agreed to disagree.

24 Google has withdrawn its offer to consent to an
25 amendment of the infringement contentions.

1 And both parties have agreed to simply reserve their
2 rights for later; to tee it up, if necessary, with your Honor.

3 **MR. PETERS:** That's right, your Honor. We have
4 agreed to disagree.

5 **THE COURT:** Well, all right, but you remember what I
6 said, you know. Shifting sands, and all that of that.

7 **MR. PETERS:** I know, your Honor.

8 **THE COURT:** If you want to stand on your shifting
9 sands, and they get shifted out from under you, then you just
10 lose. End of story. So I'm not saying you will, but we've got
11 those disclosure rules for a reason. And if your -- if your
12 disclosures aren't good enough, there won't be a second chance.
13 You just lose. All right.

14 **MR. BABER:** And I believe the last one --

15 **THE COURT:** What's next?

16 **MR. BABER:** -- your Honor, was actually not an
17 agreement, but simply to report to the Court we discussed a lot
18 of other issues in the conference: Document productions,
19 processes, electronic discovery. We didn't have a chance to
20 address them all, or even resolve many of them, but we think,
21 given the processes we've now put in place for conferences and
22 communications, we're hopeful we will be able to address those.
23 We have some disconnects in terms of your Honor's
24 order on 30(b)(6) topics, for example; but we think that with a
25 little further work, we may be able to get those resolved so

1 that we can keep continuing forward. We've started taking
2 depositions and --

3 **THE COURT:** Discovery cutoff is in July, so you'd
4 better get cracking.

5 **MR. BABER:** We're moving, your Honor.

6 **THE COURT:** Well, moving is not -- I mean, okay.
7 You've got to get moving, but there's a deadline.

8 **MR. BABER:** We understand.

9 **THE COURT:** All right. Okay. Is that it?

10 **MR. PETERS:** Yes, your Honor.

11 **THE COURT:** You all should be working for the
12 State Department, you did such a good job. You found a way to
13 just find ways to agree; not really an agreement; but you're
14 finding ways to agree. So I need you over, you know, in the
15 Mideast, I think, to help solve some problems. Okay. Thank
16 you.

17 **MR. PETERS:** Thank you, your Honor.

18 **THE COURT:** Now we're going to go to -- well, the
19 main event, which is -- and I do need that -- I've got to be
20 chairing a meeting at 3:00 p.m., so I only have 55 minutes.
21 And I'm sorry for that, but each side gets 30 minutes. So you
22 use the 30 minutes as you wish.

23 Now, I don't want -- if you want to go over into
24 arguments about the claim construction and all -- a little bit
25 of that's okay, but right now I'm basically interested in

1 learning your take on the underlying technology.

2 It does help me in these tutorials to know roughly
3 where the old technology started or stopped, and new tech in
4 the invention began, because sometimes I go away from these
5 tutorials thinking, "My God. That's great," and then I learn
6 that all of it was prior art.

7 So it helps to know where the prior art stops and the
8 new invention -- but you know, overarching thing is I want to
9 learn the basic technology.

10 So, Mr. Jacobs, you go right ahead.

11 **MR. JACOBS:** That's our goal, your Honor. And we
12 have for you the materials that we had prepared. I will
13 confess that they are the materials from the longer version of
14 the tutorial --

15 (Whereupon a document was tendered to the Court)

16 **MR. JACOBS:** -- from the longer version of tutorial
17 that we had prepared. So there's more content that we've
18 handed you than we're going to present today.

19 Our agenda is to cover the basic problem that Java
20 solves: Application development and distribution for
21 heterogeneous or diverse architectures.

22 The Java solution to that -- what's referred to as
23 "Write once; run anywhere."

24 I want to explain a little bit about the Java
25 programming environment that's background for the patents we're

1 talking about.

2 I want to spend a minute on Java licensing models.

3 We'll talk about and illustrate the patents and the
4 embodiments in the specification for three patents. And I'll
5 explain how we'll do that.

6 And then briefly I want to draw the Court's attention
7 to the basic architecture of Android, as exemplified by
8 Android -- the Android website.

9 So the problem is that computers are different. They
10 present different functionalities. They have different
11 instruction sets. The term of art is "architecture." They
12 present different architectures. And they especially present
13 different architectures to application programmers.

14 And that means that the programmer, when he -- when
15 the programmer's considering, "Am I writing this application
16 for a PC? Am I writing it for a Mac," he might write it
17 differently. He might compile it into object code differently.
18 It's all going to depend -- in the prototypical case, it's
19 going to depend on the target's architecture; the target
20 environment.

21 And so we've illustrated that here with an old Mac,
22 an old PC, because in the '90s this was a very substantial
23 issue, and an issue that the Java developers were aiming to
24 solve.

25 The solution is -- this architecture is illustrated

1 on this Slide 3, in which the programmer writes an application
2 program in, typically, Java programming language. That is
3 compiled by a compiler at the development site into something
4 called "bytecode," which is an intermediate form of code. It's
5 architecture neutral. It is not targeted at a specific
6 hardware platform or operating system. It is aimed at,
7 instead, some code that is residing on the computer called "the
8 Java Virtual Machine."

9 And so the bytecode and the Java Virtual Machine are
10 the key interface element between the code that has been
11 distributed by the application programmer, and the target
12 hardware platform. You can think of the virtual machine as
13 another layer; an insulating layer that insulates the
14 application programmer from the underlying hardware and
15 operating system.

16 **THE COURT:** May I ask you a question?

17 **MR. JACOBS:** Mm-hm.

18 **THE COURT:** If you start down there at the bottom of
19 that slide --

20 **MR. JACOBS:** Right.

21 **THE COURT:** I think I know the answers to most of
22 these questions, but those three computers -- you're not
23 claiming that is the invention?

24 **MR. JACOBS:** That's right.

25 **THE COURT:** All right. And then the physical

1 machine -- you don't claim that as the invention?

2 **MR. JACOBS:** That's right.

3 **THE COURT:** Nor the operating system?

4 **MR. JACOBS:** That's right.

5 **THE COURT:** How about Java Virtual Machine?

6 **MR. JACOBS:** The specific inventions are a product of
7 this solution. We do not have a patent that we're asserting
8 here on this whole architectural solution, but in order to
9 understand what the patented inventions arise out of, you have
10 to -- one needs to understand that the -- that this is a
11 solution, but it created its own problems. And those problems
12 are resolved by the patented inventions.

13 **THE COURT:** All right. One other question. So I
14 understand what you say. So you're saying that those four
15 blocks at the top, counting the arrow -- that, while that is
16 not the invention, the particularized inventions arise out of
17 that: Those four blocks.

18 **MR. JACOBS:** Exactly.

19 **THE COURT:** All right. Now, was virtual machine in
20 the prior art, or was that something that was invented by this
21 invention?

22 **MR. JACOBS:** There were virtual machines in the prior
23 art.

24 **THE COURT:** Okay. That helps me a lot.

25 Okay. Please continue.

1 **MR. JACOBS:** Sure. And, just to put this in the
2 everyday experience, from the standpoint of everyday
3 experience, you may have noticed, sitting at a computer one
4 day, that code asked for the latest version of Java on your
5 desktop machine. And that's exemplified here by this little
6 Java coffee cup.

7 What happens then is your computer goes to a
8 Java-delivering website, and it delivers the latest version of
9 the Java Virtual Machine, or, as it's often referred to, "the
10 Java runtime environment"; the "JRE." So that's how we
11 encounter and deal with Java in our daily life on computers.

12 So the Java programming environment consists of the
13 Java programming language. And the Java programming language
14 is a programming language. And it's documented in a book
15 called, The Java Language Specification.

16 And it's an object-oriented language, which means
17 that it is -- that the code is written in packages. Those
18 packages are called "classes." So the class is the key
19 construct for a Java programmer. And the class is a -- can be
20 thought of as a template or a form for instances of the class
21 which will be objects.

22 So in the case of an e-mail, for example, the class
23 will define the various fields of your e-mail message: Your
24 "To"; your "From"; your "Subject" line. And then it will have
25 methods that will cause the e-mail message to be sent, read,

1 and received, et cetera.

2 **THE COURT:** All right. Let's pause on that.

3 Coming into today's hearing, I could not understand
4 exactly what was meant by "class" -- that term. So go over
5 that again, and help me understand what -- how "class" is used
6 in this -- in this technology.

7 **MR. JACOBS:** So as I understand it, literally, when
8 you are programming in Java, you say to yourself, "I'm going to
9 divide this overall program into a bunch of tasks and
10 definitions, et cetera." And you divide that into what are
11 called "classes."

12 In the old days, preJava, you programmed in modules.

13 In Java, you program into a class.

14 And the basic difference is that a class keeps the
15 data and the executable code -- if you will, the methods --
16 together, and encapsulates them with clearly defined interfaces
17 between classes. That's the essence of object-oriented
18 programming.

19 And so the class is -- you literally write "class."
20 And then you write the code and the data definition under it.

21 And, as you'll see -- I think this next slide will
22 help. What happens, then, in the Java environment is that the
23 programmer has organized his or her source code into classes.
24 The compiler has compiled that into bytecode. And the bytecode
25 itself is a class file or, more precisely, a .class file.

1 And then it is those .class files that the virtual
2 machine -- the format of which the virtual machine understands,
3 and can execute on the target platform.

4 Now, the reason I think the meaning of "class" is
5 somewhat confused is that classes are just sort of pervasive in
6 Java. And, in particular, there's something called "the Java
7 class libraries." And those are libraries of classes. So
8 those are libraries of code that have been written and
9 distributed, with the idea that, rather than having an
10 application programmer have to rewrite that code each time, the
11 application programmer can just write an interface to that
12 code, and invoke it at runtime.

13 So the class is something that's written by the
14 programmer. It's also something that is available in the Java
15 Platform. And there are dozens and dozens of these classes in
16 the Java class library.

17 **THE COURT:** This one said "class calendar." What --
18 what would that be?

19 **MR. JACOBS:** Exactly. So that is a calendar
20 function. In the e-mail example, perhaps you want it to get
21 the date, so you -- rather than writing code for a calendar,
22 you invoke the calendar function through the application
23 programming interface of that class. It's a function, but it's
24 been -- it's in the form of a class. And on the runtime in
25 the -- on the computer, the end user's computer, those class

1 libraries are again available. Just as the bytecode was
2 deposited in .class files, the class libraries are available at
3 runtime or execution time for the application program to invoke
4 and cause to be executed.

5 And those class libraries have an application
6 programming interface which is documented in a couple of books
7 called The Java Application Programming Interface, and, more
8 recently, on the Web in documentation, so that programmers know
9 how to write the application program interface -- how to use
10 the API when they're writing their code.

11 **THE COURT:** What were those books that you just held
12 up?

13 **MR. JACOBS:** This is actually from about '96, I
14 think. This is The Java Application Programming Interface,
15 Volume 1 and 2, by Gosling, Yellin, and the Java team.

16 **THE COURT:** What's the other book?

17 **MR. JACOBS:** Hm? It's Volume 1. Sorry.

18 **THE COURT:** Oh. That's Volume 1.

19 **MR. JACOBS:** Volume 1 and Volume 2.

20 And then the other book is The Java Language
21 Specification, by Gosling, Joy, and Steele.

22 **THE COURT:** Okay.

23 **MR. JACOBS:** So Java licensing models. The -- Java
24 has been in existence now for -- commercially, for around 15
25 years. And Sun, of course, commercialized Java, and Oracle

1 inherited that.

2 It's a quite complicated commercial licensing scheme,
3 but boiled down to it, there are really three ways to get Java
4 from Sun/now Oracle. One is through royalty-bearing commercial
5 licenses. If you want to distribute Java on your system, then
6 you can enter into a commercial license with Oracle. And there
7 are -- these terms are all over the Java industry. The SCSL or
8 the TLDA are well-known acronyms for Java licensing.

9 An important part of Sun's strategy for Java was to
10 penetrate the academic environment and get students programming
11 in Java. It was enormously successful. Java is the leading
12 programming language in universities today. That was
13 facilitated by a royalty-free academic licensing program.

14 Then most recently, Java was released by Sun --
15 what's called "OpenJDK." It was released under the GPL; the
16 new public license. The GPL is a very strong open-source
17 license, in the sense that, if you are a recipient of code
18 under the GPL, you inherit GPL obligations when you modify or
19 extend that code.

20 So, just as the original source -- in this case,
21 Oracle -- has to make available GPL code as open-source code,
22 the recipient of that code must similarly make his or her code
23 on top of the Oracle code available as open-source code, and
24 all the way down the line. That has the effect, probably, of
25 inhibiting its commercial use.

1 There are some commercial uses of OpenJDK, but it has
2 not had a substantial impact on the commercial licensing
3 business of Oracle, because of these strong GPL restrictions
4 that are imposed on downstream users.

5 **THE COURT:** The word "open" --

6 **MR. JACOBS:** Open source.

7 **THE COURT:** What does that mean?

8 **MR. JACOBS:** Open source. The source code's publicly
9 available.

10 **THE COURT:** Without fee?

11 **MR. JACOBS:** Without fee; but with this copy left, as
12 it's referred to, obligation to make your own source code
13 publicly available.

14 So what will happen is, when we talk about Android
15 later in the case, what you'll hear is that Android did not
16 adopt OpenJDK, because the GPL restrictions were viewed as too
17 constraining on handset developers. That's why this is
18 ultimately going to be relevant.

19 **THE COURT:** Okay.

20 **MR. JACOBS:** So let's talk about the patents-in-suit,
21 and, more particularly, three patents that present
22 claim-construction issues for the Court in a couple of weeks.
23 There are actually six patents that raise issues; but three of
24 those only raise this computer-readable-medium issue that
25 you'll be getting into when you read the briefs. Three of them

1 raise more technical issues, if you will. And those are the
2 '104, the '702, and the '520.

3 So the '104 patent is the earliest patent in the
4 case. It dates back to the very early days of Java
5 development. And it is about resolving data references in
6 generated code.

7 And, in order to understand this, I need to step back
8 once more into this issue of diverse architectures and the
9 challenge facing application programmers. So an application
10 programmer can program in a source language, and then compile
11 that code into a form that is intended for a specific machine.

12 If you are writing for a SPARC workstation, then you
13 would take your source code, run it through a Sun compiler that
14 is designed for the SPARC environment; similarly, for an IBM PC
15 with an Intel chip. And then the program would execute that
16 object code tailored for that target environment. Indeed, it
17 was optimized for that target environment. So we're talking
18 about code that should execute, if well written and the
19 hardware can support it, lickety split.

20 This is the classic way that programs were
21 distributed. If you and I went to Egghead Software 20 years
22 ago to buy software for our personal computers, we looked at
23 the side of it, and it said, "This will run on a Windows
24 machine with a 386 processor," or "This will run on a Mac."
25 That code went through a compiler intended for that

1 environment.

2 The other way to execute code is through an
3 interpreter. And in the interpreter case, the source code is
4 actually installed on the end user's machine. And an
5 interpreter interprets each source instruction, one by one, and
6 executes the code.

7 This interpreter is flexible, in the sense that all
8 you can take is source code. And source code is relatively
9 portable, relatively usable, machine to machine, but it's slow,
10 because you're going through this interpretive step, and this
11 step-by-step execution process.

12 So the '104 patent explains that what they are about
13 in 1992 is developing a hybrid compiler/interpreter, comprising
14 a compiler for compiling source code, and an interpreter for
15 interpreting the compiled code. That's the backdrop to the
16 invention of this patent.

17 An example of that is the Java architecture, as
18 illustrated here in the way that we've illustrated earlier.

19 So the patent sets up the basic problem here, by
20 explaining the following.

21 In Figure 1A, we're dealing with a compiler model.
22 And we've taken some source-level instruction that required a
23 variable to be captured. And the compiler, when interpreting
24 that source-level instruction, figured out in the compilation
25 model is the way to do that is do a Load 2 instruction, and

1 find the data value at Slot 2.

2 So compilation gets all the way down to the
3 data-structure level, and tells the computer, "Go to Slot 2,
4 and find the value we're looking for." It's fast, but it's not
5 very flexible.

6 The other model is interpretation. And the '104
7 Patent discusses what happens under interpretation.

8 **THE COURT:** We have it there now.

9 **MR. JACOBS:** That's in Figure 1B. And in Figure 1B,
10 we've -- the interpreter is receiving the instruction in the
11 form, "Load Y." It's an intermediate form of an instruction
12 that was probably something like, "Add X and Y," or "Draw a
13 point, X and Y." That got compiled to an intermediate
14 instruction of the form "Load Y"; but now the interpreter has
15 to -- each time it encounters this instruction, it has to go
16 figure out where Y is. And that is going to take machine
17 cycles. That is going to be slow.

18 Now, recall that what we're going to do in Java is
19 we're going to install this virtual machine layer on our
20 end-user computer. And unless we change it, it's going to
21 going through these steps. It is going to receive bytecode.
22 It's going to encounter "Load Y," and it's going to resolve it,
23 time after time. And that is going to be slow and inefficient.

24 So what the inventors came up with in 1992 was a kind
25 of a hybrid solution for this hybrid architecture. And the

1 hybrid solution is to resolve this symbolic reference -- the
2 "Load Y" reference -- the first time; but the next time around
3 and subsequent times, rely on that previous resolution to the
4 numeric value.

5 So we --

6 **THE COURT:** Say that again; that last sentence.

7 **MR. JACOBS:** It's going to rely on the resolution to
8 that numeric value. So we'll see that the -- in this
9 illustrative example in the patent, in the embodiment
10 disclosed, the first time the interpreter goes through this
11 process, it will find Y. It will find that it is in Slot 2.
12 And then it will, somewhere, whether -- and this is part of the
13 issue between the companies. Whether it's by literally
14 rewriting the old reference, or somewhere else, some other
15 method of relying on the fact that this resolution has
16 occurred, the next time around, it will go straight to Slot 2.

17 So the symbolic reference has been understood once,
18 and then, in a sense, converted to a numerical reference,
19 speeding up execution. And this was a quite important
20 contribution to the art of this hybrid compiler virtual machine
21 architecture.

22 And we can see this expressed -- the invention
23 expressed in an illustrative claim: Claim 11. So we have
24 intermediate form object code. That's what the compiler
25 generated in this hybrid architecture. And it has symbolic

1 references. In the illustration, that's a Y.

2 And then we are going to determine a numerical
3 reference that corresponds to the symbolic reference. And then
4 we're going to obtain data in accordance with that numerical
5 reference. So I think of this. In my own mind, this is
6 hybridizing the hybrid architecture, because, at the virtual
7 machine, we're doing a kind of a compiler-like thing after
8 we've received the code. We're resolving the variable into a
9 numerical reference, doing it once, and not doing it
10 repetitiously.

11 So the next patent with some substantive
12 claim-construction issues is the '702 class file redundancy
13 removal. So this is why it's important to understand class
14 files. So the -- because the problem that this patent
15 addresses is that, as we explained, we have the class files
16 written by the application programmer. They retain their
17 "class fileness" after compilation by a Java compiler. They
18 are now .class files.

19 And then we have the Java class libraries, which are
20 also class files.

21 And the upshot of all of this is that the -- when --
22 if you just load all of this up in memory and don't do anything
23 else to it, as the patent explains, there could be hundreds and
24 thousands of class files taking up space in your memory. And
25 so what the '702 patent discloses is a way to solve that

1 problem. And the way that is going to be occur is by taking
2 out the redundant elements, and putting them somewhere else.

3 So we have some illustrative class files here. The
4 different colors represent different kinds of different
5 elements.

6 It could be a lengthy string that would otherwise
7 take up a lot of memory. And what the -- what a preprocessor
8 does is examine the class files; pull out the common elements;
9 put them in a shared location; and create a new format called
10 an "mclass," or modified class files. The original class files
11 have now been compacted by the removal of the shared elements.
12 And illustratively, we're now using far less memory space.

13 Now, this is a -- an important function, because, as
14 I mentioned, the Java bytecode has all of these class files in
15 it that are independent chunks of code. This results in part
16 of this being an object-oriented programming language.

17 Now, there's another step that's worth noting,
18 because this is kind of a technical point that goes to the
19 claim construction -- claim-construction issue here. In the
20 original class file, recall the class files are self-contained.
21 So the data and the code that relates to it are all in the same
22 little package. And so the references in the original class
23 file to what will ultimately be defined to be a shared
24 element -- those are internal references.

25 When the mclass file is created, this new package

1 that contains the reduced class file and the shared elements
2 that has to be -- somehow the reduced class files need to be
3 adjusted, so that they point to the shared location. They're
4 not merely -- technically, they can't just be a subset of the
5 original reduced class files. They have to be modified or
6 adjusted in order to point now to this shared location.

7 And, in fact, the specification discusses this, by
8 noting that the -- there's a new constant type. And that new
9 constant type replaces the duplicated element in the embodiment
10 disclosed and discussed here, and replaces it with this
11 pointer; this index into the shared location. So that's the
12 '702.

13 An illustrative claim. We're going to preprocess the
14 class files. So this is before the file is actually loaded
15 into the virtual machine. We're going to form a shared table.
16 We're going to remove duplicated elements. And then we're
17 going to form this multiclass or mclass file that has a
18 plurality of the reduced class files and the shared table.

19 The last patent that we'll be discussing is the '520.
20 And Ms. Agrawal will explain that patent.

21 **THE COURT:** Thank you, Mr. Jacobs.

22 **MS. AGRAWAL:** Good afternoon, your Honor.

23 **THE COURT:** Good afternoon.

24 **MS. AGRAWAL:** So we talked with you earlier about how
25 Java source code gets compiled into bytecode.

1 And what the inventors of the '520 patent realized
2 was that sometimes the bytecode -- certain functions expressed
3 in bytecode can be inefficient as is. So what the inventors
4 realized is that you start off with the source code; compile it
5 into bytecode. The inventors realized that if you simulate
6 execution without execution, or simulate --

7 **THE COURT:** Say that again. You simulate?

8 **MS. AGRAWAL:** You simulate execution before actual
9 execution. You play execute it.

10 **THE COURT:** Uh-huh.

11 **MS. AGRAWAL:** You go line by line, and you simulate
12 the execution of this bytecode. You figure out what its
13 operation is, and you determine a reduced set of instructions.
14 A reduced set of instructions is about -- it's much fewer lines
15 of code than the original bytecode. And the reduced set of
16 instructions get fed into the virtual machine for actual
17 execution.

18 **THE COURT:** Is that done every time you run the
19 program, or just -- this is a designer tool?

20 **MS. AGRAWAL:** This is designed by Java engineers.
21 The idea is that it's done for certain sorts of static
22 initializations.

23 The primary example discussed by the '520 patent are
24 static initializations of erase, because they come up often.
25 And the inventors realize that the way that the bytecode

1 instructions that was designed, there's this -- there's an
2 opportunity to offer this optimization.

3 **THE COURT:** Well, let me rephrase my question.

4 If I were to go out and buy a modern computer with
5 modern software, and -- does it have this play execution
6 featured on it, so that each time I run my software, it figures
7 out a more compact way to economize on steps? And -- or -- you
8 see what I'm saying? Is that the way it works?

9 **MS. AGRAWAL:** Yeah. So it's not observable to the
10 end user, in the sense that you don't -- that you don't
11 actually see --

12 **THE COURT:** I don't see it, but it's going on inside
13 the computer?

14 **MS. AGRAWAL:** But it's going on.

15 **THE COURT:** Okay. Good. All right.

16 **MS. AGRAWAL:** And here's our illustrative claim of
17 the '520 Patent, which just covers exactly what we've talked
18 about. You receive bytecode, and you play execute the
19 bytecode, without actually running the code, to identify the
20 operation, in order to figure out what it does, in order to
21 create a reduced set of instructions.

22 **THE COURT:** Let me just read that.

23 A method comprising. Receiving the code. Play
24 executing the code without running the code. Among the
25 processing component to identify the operation. If the code

1 were run by the processing component and creating an
2 instruction for the processing component to perform the
3 operation.

4 Okay. All right. Go ahead.

5 **THE COURT:** Is that it?

6 **MR. JACOBS:** So these are the -- where we started
7 with was this point that these inventions arise out of the Java
8 architecture. They solve problems created by this solution to
9 the problem of distributing code for differing computers.

10 It's -- the Java architecture is worth -- it's worth
11 observing how the Java architecture is exemplified in Android.
12 Just to set up where we're going in this case, this is all
13 material that is available on the Android website.

14 So if you go to the Android website, you'll see that
15 the application programming interface level, the Android
16 software development kit provides the tools and application
17 programming interfaces necessary to begin developing
18 applications on Android using Java.

19 So what is the programming language for Android?
20 It's the Java programming language.

21 And then this virtual machine architecture is --
22 is -- has its modeling in Android. Every Android application
23 runs on an Android device under a virtual machine. In Android,
24 that virtual machine is called "Dalvik." And it runs classes
25 that have been compiled by a Java language compiler. And then

1 Android developers added this step of converting the Java
2 bytecode into something called "dex," or ".x," using the dx
3 tool.

4 And Android includes a set of core libraries that
5 provide most the functionality available in the core libraries
6 of the Java programming language. So we talked about
7 libraries; the class libraries. A subset of class libraries
8 are called "core libraries." And Android provides a set of
9 those.

10 So one of them is this calendar application
11 programming interface. We saw this earlier in the Java
12 documentation. And this is a set of the Java core libraries at
13 Slide 28. These are Java core libraries provided with the
14 Android distribution.

15 So if you go look at the Android source code, you
16 encounter this set of java.awt.font, et cetera, and java.beans,
17 and down the line. This is basically a compare of the Java
18 core libraries with the Android core libraries.

19 And so if you look at the Java architecture and the
20 Android architecture, you see that in both cases, the
21 programmers are programming in Java. They're both -- in both
22 cases, they're presented with class library APIs. Actually,
23 both of them use a Java compiler. In the case of Android,
24 there's this add-on of Java bytecode translated to dex code.
25 They both present this virtual-machine architecture, so that

1 the bytecode can run on any device that has the VM installed.
2 They both present Java runtime -- Java class libraries at
3 runtime. And then, of course, it can be any operating system
4 and physical machine that will support the virtual machine in
5 each case.

6 And our basic contention is that, having chosen that
7 architectural similarity, what we'll be proving later in the
8 case is that they didn't have much choice but to implement the
9 inventions embodied in the patent-in-suit.

10 **THE COURT:** Now, the -- is there a copyright claim in
11 this case? I thought there was.

12 **MR. JACOBS:** There is a copyright claim.

13 **THE COURT:** Just on that slide -- I know we're here
14 today on claim construction, but how does the copyright claim
15 figure into that slide?

16 **MR. JACOBS:** It figures into the slide here
17 (indicating). It figures into the implementation of Java APIs;
18 documented and copyrighted documentation made available under
19 license that has requirements that Android does not meet. And
20 it figures in the Java runtime libraries which implement that
21 API.

22 And then there are some elements of actually directly
23 copied code that we have observed in the two sets of the code
24 represented by each implementation.

25 **THE COURT:** All right. Okay. Thank you. You've got

1 about two minutes left for last word at the end, but let's now
2 give equal time to the defendant, Google.

3 **MR. WEINGAERTNER:** Good afternoon, your Honor.

4 **THE COURT:** Good afternoon.

5 **MR. WEINGAERTNER:** We also have some hard copies.

6 (Whereupon a document was tendered to the Court).

7 **THE COURT:** I only need one of these, right?

8 **MR. WEINGAERTNER:** Yes, your Honor. I can take those
9 back.

10 **THE COURT:** Give the rest to my law clerk.

11 **MR. WEINGAERTNER:** In the months that have gone by
12 since filing this lawsuit, Android devices, which are accused,
13 have become the fastest-selling smartphones in the world.

14 Oracle's point here is that that is somehow due to
15 Java technology.

16 The Court, in its scheduling order, pointed out that
17 this dispute is a Java-fueled dispute, and that is certainly
18 true. Our view is not that it's a Java-fueled product, and
19 we'll explain that to some extent.

20 Our to tutorial is, I think, focused on a little bit
21 more technical nuts and bolts. I know we don't have a whole
22 lot of time. I'll try to go through them in a meaningful way.
23 And perhaps if your Honor has specific questions, I'm happy to
24 try to address them to the extent we can. Obviously, the
25 subject matter is very technical. The seven patents-in-suit --

1 this, I think, transcends claim construction, to try to give
2 the Court more tools to be able to understand all the different
3 balls that are in the air in this case.

4 And what I'll try to do is tie them up at the end,
5 and perhaps provide a little bit of rebuttal to some of the
6 things that Oracle's counsel pointed out.

7 Next slide, please.

8 What I thought might be helpful is first to give just
9 kind of a brief overview to provide some context for what's
10 happening on the accused-product side.

11 Google and other members of an organization, a
12 nonprofit-type organization called "The Open Handset Alliance,"
13 made up of various companies, including handset developers,
14 developed Android as an open-source platform.

15 Your Honor asked about what that means. In this
16 case, it means truly free. I think there are probably
17 differences of view as to whether it is economically free. In
18 this case, Java is free both monetarily, and also free from
19 basically restrictions on how it's used.

20 There is a compatibility test that people can do, and
21 then they'll use the Java trademark; but it's very open. And
22 when -- and what I'll do is point up to the screen here.

23 The Android open-source code is provided on a server.
24 Anybody can access it, but it's typically accessed by folks who
25 can actually put it to good use, such as handset and device

1 manufacturers. And they create devices like the one sitting
2 right here on my belt. And a lot of folks in the room have
3 them as well. Some of the better-known ones are Motorola,
4 Samsung, HTC.

5 And that code then gets downloaded onto a handheld
6 device. And the folks that create the code that actually gets
7 on the device can do basically whatever they want with it. And
8 that's actually a bit of an issue in the case. It relates to
9 whether the burden is being met on infringement contentions.
10 So it doesn't really relate so much to -- it relates a little
11 bit to claim construction, because it has to do with how the
12 claim language is parsed and applied throughout what is sort of
13 known affectionately as "the ecosystem."

14 And so the ecosystem, again, starts with the creation
15 of the open-source code; flows down through the handsets.
16 Handsets are generally sold by the mobile operators, who give a
17 discount because you, you know, sign up for, you know, a
18 contract of some kind to use their services. And then there's
19 developers that create applications.

20 And the idea behind -- one of the ideas behind open
21 source and behind Android is that, by having this be free,
22 application developers flock to the opportunity to create code.
23 And there are architectural features Android that have made
24 this specifically attractive. They, I don't think, have much
25 to do with this case. I think they're independent.

1 Android is a big project. I think there are
2 something like 11 million lines of code over many years. It's
3 been, again, developed by a number of different folks; Google
4 being sort of the lead shepherd.

5 And the developers have access to something called
6 the "Android SDK." It's the development kit. And they create
7 applications. And anybody can do this. Big companies do it.
8 Individuals do it. I can go onto a store and download an app.
9 I can write my own. I can reach in and modify code. One of
10 the folks in our legal team is the type of person who does
11 that. I haven't done it myself yet.

12 Next slide, please.

13 This just gives kind of an overview. I know that
14 some of the typeface is small, your Honor, so there's no way to
15 really fit it in otherwise. And this is kind of a map of the
16 Android architecture. Other products, including Java, you
17 know, have illustrations that kind of show this sort of
18 "stack," as it's called.

19 And the different layers are sometimes known in
20 computer science -- known as "layers of abstraction," in some
21 sense.

22 At the top are applications that we're all familiar
23 with: Keeping our contacts, you know; an actual phone
24 functionality browser.

25 There's the application framework. That is a little

1 bit more difficult to parse.

2 "Window Manager" is sort of ascertainable as we stand
3 here in the court.

4 And there are libraries, which are the next level of
5 kind of complexity in a way that are harder to apprehend as we
6 sit here today.

7 And on the very lowest level is something called "the
8 Linux kernel." Linux is an operating system developed -- I
9 think it's some sort of derivative of UNIX in some way, that
10 was developed as open source probably 20 years ago or so, or
11 maybe a little bit less, that provides a lot of underlying
12 tools that people commonly rely on.

13 Now, where the action is in this case is the Android
14 runtime environment. And within that -- and I apologize,
15 your Honor, that the typeface is small here, but it's the
16 Dalvik virtual machine, which is named for a town in Iceland
17 where one of the engineers' family used to live, for whatever
18 that's worth.

19 And there are core libraries that are specific; that
20 are very tightly wound up with how the Dalvik virtual machine
21 works, that are kind of shown here.

22 And the rest of this refers to and calls
23 functionality; run -- running applications that people have
24 downloaded.

25 Next slide, please.

1 We thought it would be helpful to give just a very
2 kind of high-level overview of just what the structure is of a
3 computer as you sit there and look at it; not --

4 **THE COURT:** I understand that much. Skip that one.

5 **MR. WEINGAERTNER:** Okay. Thank you, your Honor.
6 Appreciate the feedback.

7 Just a brief word on APIs. This was something that
8 your Honor asked about. One of the things -- and I'll just go
9 very briefly on this. It's basically a mechanism to allow
10 different applications, different programs, to talk to one
11 another. It makes -- it makes the market function more
12 smoothly, and allows programs to reach down into things like
13 libraries, which we referred to earlier -- or Mr. Jacobs
14 referred to earlier -- without everybody having to -- it's sort
15 of a Tower of Babel solution, in a way, your Honor.

16 And just one quick note is that APIs permit an
17 application to reach down into the operating-system layer or
18 into other programs.

19 I thought it might make sense to be speak very
20 briefly about just what programming languages are, and how they
21 differ.

22 There's very high-level languages, which are sort of
23 human-readable. Folks who write code look at them as being
24 very human readable; but machines can't read them. So you have
25 to kind of move from the human write-able and readable side,

1 down to a form that ultimately, as your Honor, I'm sure, knows,
2 ones and zeroes that represent high and low voltages that
3 computers use to actually carry out operations.

4 The translation from one level to the other that
5 needs to be done is done by compilers that take the kind of
6 human-readable code, and takes it down to a level that's
7 general not human readable, although some of our engineers say
8 that they actually can read some of this very low-level code.

9 **THE COURT:** Literally the ones and zeroes on the
10 right side there?

11 **MR. WEINGAERTNER:** Yes, your Honor.

12 And, in fact, I'm glad your Honor raised that
13 question, because there's an issue in this case of what it
14 means to be in an intermediate representation.

15 This case involves something called "bytecode,"
16 which, is in a way, in between high-level languages. It's a
17 little bit easier to see what's going on. And bytecode is
18 important for how virtual machines operate. And I have a few
19 slides on that.

20 So a virtual machine can be thought of, in a way, as
21 an adapter that allows -- the adapter is basically written for
22 each piece of hardware operating system. And once it's in
23 there, you can write code that runs on any of them. And
24 Mr. Jacobs referred to that. It's an old concept that goes
25 back at least as far as the '60s. And there are reasons why it

1 became popular later, but it's -- the idea here is that
2 somebody does all the work for you. Somebody writes the
3 virtual machine --

4 **THE COURT:** That right-hand column, you're saying,
5 goes back to the 1960s?

6 **MR. WEINGAERTNER:** Yes, your Honor. And we've got a
7 time line.

8 **THE COURT:** We'll come to that.

9 **MR. WEINGAERTNER:** Okay. I --

10 **THE COURT:** Just walk me through particularly that
11 second and third step there on the right-hand column.

12 **MR. WEINGAERTNER:** Yes, your Honor.

13 And so -- and again, we've got user here, but in a
14 way, you can also imagine a developer being there who writes
15 the code. The code is written in a way that -- using words
16 like "If," Then," Fetch," Execute," perhaps, or other -- or "Do
17 this for this period of time until this condition is met."

18 And that kind of -- if you reduce it down to the ones
19 and zeroes level, it has to be very specifically done for this
20 operating system. So the virtual machine says, "Hey. You know
21 what? Just compile it down a little bit; not fully down to --
22 but somewhere in between."

23 And that virtual machine will then take that next
24 step from that language down to a specific operating system.
25 So again, the folks that write the virtual machines who do the

1 hard work of building the bridge here -- that's very specific
2 to the operating system which is, in turn, built to run on a
3 particular hardware.

4 **THE COURT:** Yeah, but over on the left-hand side,
5 you've still got it running with ones and zeroes at the bottom
6 with the orange box. So what's the difference between that
7 left column and the right column?

8 **MR. WEINGAERTNER:** That's a great question,
9 your Honor. The difference is that this native application
10 right in here, which we don't show, is a compiler. Compiles
11 down specifically for this machine.

12 And so what I have to do is actually write this
13 application for this operating system. And so if I'm an
14 application developer, it's kind of a huge burden for me to
15 say, "Well, I'm going to write one to run on Mac; and write
16 another one to run on Windows machine; and write another one to
17 run on Android," or whatever it might be.

18 So the reason that this wasn't always done, because
19 it seems sort of -- well, why would we always do this? -- is
20 because it involves more processing time. And that was an
21 issue. Your Honor may recall there was a time when computers
22 ran very slowly, and they didn't have much memory. And so
23 people would use these things, but it was a real burden.

24 And in the '90s, as you know, chips became a lot
25 faster. And we all can see the effects of that around us.

1 Virtual machines became more practicable.

2 And, in fact, the advent of the Internet made them
3 more practical, in a way, as well, because you'd have code that
4 would be coming down over the 'net to any device. And so --
5 and so Java, in a way, kind of rode the wave. And, in fact, we
6 have a quote later in the presentation that your Honor can read
7 if we don't get to it, that kind of credits, in a way, the rise
8 of the Internet with Java. And Java kind of caught that wave,
9 in a way, and rode it; but the concept's predated by more than
10 one decade, for sure.

11 **THE COURT:** And the phrase "virtual machine" was back
12 in the '60s, too?

13 **MR. WEINGAERTNER:** Yes, your Honor.

14 Next slide.

15 Just an example of some source code; nothing all that
16 interesting, but I guess the thing, I suppose, is it's slightly
17 more human readable.

18 Just flip back for a moment.

19 There's actual English typed words in there. This is
20 the most simple of all programs. It basically says, "Print to
21 the screen. Hello world."

22 It's used a lot in textbooks to show this basic
23 structure. And this is called, and it doesn't return a value.
24 It just says, "Print to the screen," but that gets represented
25 at the machine level in this language that only computers can

1 read, where -- sort of that's designed for computer reading.
2 That basically allows the microprocessor that -- most of which
3 had been developed down the road here -- can operate on. So
4 the compiler basically translates from one to the another.

5 Actually, probably skip this one (indicating).

6 Again, the point here is that you can't run on two
7 different systems. That was -- hearkens back to the point
8 your Honor raised earlier: What was the difference?

9 Well, here, there is no compatibility. We all
10 remember the days when you agonized over which computer to buy,
11 because you knew that you might love a Macintosh or you might
12 love a Windows machine, but some of them, you couldn't run
13 certain programs and certain games on them, for example.

14 Next, please.

15 So again, just to cover virtual machine -- and we
16 have other definitions -- you know, it basically simulates a
17 physical machine; simulates an operating system, so the
18 programs don't have to worry about where they're running. They
19 think they're running on any machine.

20 And we can skip this.

21 Just for your Honor's reference, there's a few
22 different definitions out there. They're not too
23 controversial.

24 **THE COURT:** Still, it must take more time, I would
25 think.

1 **MR. WEINGAERTNER:** Certainly it takes more time to
2 actually run, because you have a whole additional level of
3 processing.

4 And counsel for Oracle pointed out that their -- that
5 their improvements are directed to, the way I see it,
6 efficiency and improvements. They try to make these things run
7 a bit faster. Real estate is at a premium on a cell phone.
8 Power is at a premium. You know, every little bit helps, so to
9 speak; but you know, in our view, it's a very incremental
10 invention. And it's not the whole virtual machine was
11 invented, by any stretch.

12 Okay. Next slide, please.

13 Again, the term that comes up is this "Write once;
14 run anywhere" concept. Again, that sort of goes hand in glove
15 with the concept of a virtual machine, and disconnected from
16 any hardware.

17 Next, please.

18 Just a quick note. There are fundamentally different
19 ways to realize a virtual machine. And this is a little bit
20 detailed, because it comes from stack-based architecture, where
21 the instructions are going to push down, kind of like cafeteria
22 trays. It's called "Last in, first out." You push your
23 instructions down, and then you kind of pop them back off.
24 It's a way of keeping track of where you are, where you
25 actually run each step.

1 Next, please.

2 There's also a completely different approach, which
3 is register based, which you can think of as, rather than
4 pushing down onto a cafeteria tray, you reach individual
5 cubbyholes that have specific names on them. And there's -- it
6 goes down to the fundamental --

7 And that's actually an important point here. Java --
8 the Java Virtual Machine is a stack-based architecture. This
9 isn't directly, you know, relevant to any particular claims in
10 the case, but it's the kind of important, we think, in terms of
11 telling the story, that -- that Android is register based. And
12 in some ways, that actually permits/facilitates multitasking,
13 which is something that Android is regarded as doing very well.
14 It makes the handheld devices more useable. People have
15 multiple applications running at the same time. And a
16 register-based approach facilitates that to some extent.

17 Next slide.

18 So this just compares. Our point is that they're
19 different.

20 Next, please.

21 So, your Honor, this is a time line that shows the
22 various virtual machines over the course of time. And these
23 are all, you know, searchable on Wikipedia and other places.
24 Just to kind of get a sense of where we are, Java Virtual
25 Machine, you know, falls here; a couple, three decades after

1 some of the earliest ones. Dalvik is out here. And Flash is
2 one you're probably familiar with, your Honor, that permits
3 running of videos, and so forth, on a machine. Microsoft has
4 its version.

5 So, just to go over pros and cons, programmers can
6 create a single program that will work equally well. It's
7 great for the Internet, because you don't have to are worry if
8 it gets downloaded to whatever machine happens to be running
9 the code. And, of course, folks who have code are continually
10 making it available to be downloaded to create this ecosystem
11 that we referred to.

12 The problem is that it can be slower, because it's an
13 extra layer of interpretation. It would be like walking around
14 with an interpreter by your side who had to interpret every
15 sentence that you say.

16 Another potential issue is that, because you're kind
17 of reducing to a particular common denominator, you have to
18 take into account that different machines might have different
19 features, and you have to speak kind of commonly to the ones
20 that overlap with each other.

21 Okay. This is something that your Honor raised
22 earlier. What is this bytecode?

23 Bytecode, again, is something that's a notch above
24 the actual underlying machine code, in a way. It's an example
25 of the intermediate form that code can be in. And that's one

1 of the terms that's being construed in this case.

2 We're not going to go into details on that. We just
3 wanted to make sure it was clear kind what was really at issue
4 here.

5 And again, here -- here's an example of what bytecode
6 can look like. It's a little bit more abstract. It can
7 actually be implemented in ones and zeroes.

8 And what's important, I guess, to know as well is
9 that the virtual machine doesn't compile the code down. It
10 actually interprets the code. So it goes instruction by
11 instruction, and then reduces it down to the ones and zeroes to
12 actually carry out the instruction. And that can be slower as
13 well than compiled code, that's already down at that level.

14 **THE COURT:** Can you go back to that last slide right
15 there? Right there.

16 **MR. WEINGAERTNER:** And then, just to skip ahead one
17 -- so the point here is that you can actually run Windows on
18 Apple. And the folks that actually write the virtual machine
19 did the underlying footwork to create that sort of adapter that
20 sits on each of these machines.

21 And this is --

22 In fact, just put all three of them up.

23 This just gives an example of, side by side, what the
24 differences are between source code, bytecode.

25 **THE COURT:** Let's pause on that.

1 **MR. WEINGAERTNER:** Yes, your Honor.

2 **THE COURT:** What is that language that's on the
3 bottom left?

4 **MR. WEINGAERTNER:** Okay. This one -- I think I'll
5 have to ask Mr. Francis. I think this is in Java. I'm not
6 exactly sure. Java, in a lot of ways, is actually relatively
7 similar, and actually derives in some ways from C, which is an
8 older programming language.

9 **THE COURT:** Well, okay; but I'm curious. Go ahead
10 and ask Mr. Francis.

11 **MR. FRANCIS:** Your Honor, it's simply an example of
12 what the code at the source-code level would look like. This
13 particular example in this instance, it's creating a button on
14 the screen that --

15 **THE COURT:** I understand. What's the name of the
16 language?

17 **MR. FRANCIS:** I believe it is Java, your Honor.

18 **THE COURT:** Okay. All right. So that one -- so
19 then, to get down to the bytecode level, those -- why is it in
20 different colors? I don't think that one that's in orange and
21 purple and green.

22 **MR. WEINGAERTNER:** Probably bad design choice,
23 your Honor, I have to candidly admit.

24 **THE COURT:** So the reason I ask you --

25 **MR. WEINGAERTNER:** This could easily be in black.

1 **THE COURT:** So the -- the middle step -- what does
2 that correspond to in the left-hand step, where it says
3 "iconst_0"?

4 **MR. WEINGAERTNER:** Again, this is, I think, an
5 example, your Honor. It would be very difficult, because each
6 one of these instructions will require particular operations,
7 like store a particular value. So it's actually not really
8 even possible. If you could imagine, there isn't a one-to-one
9 correspondence. You'd have to basically take one of these
10 instructions, and kind of expand that.

11 **THE COURT:** Is that a real thing, or did somebody
12 just pull out things that look nifty, and stick them up there
13 but they didn't really work?

14 **MR. WEINGAERTNER:** It's real. In terms of the
15 format, what we didn't do was take a particular instruction and
16 de-compose that. I'm not sure. I guess it's an example.

17 **THE COURT:** It's just illustrative. It may not be
18 real. All right.

19 And then the bytecode gets reduced down to zeroes and
20 ones.

21 **MR. WEINGAERTNER:** Under this -- I'm sorry,
22 your Honor.

23 **THE COURT:** Okay.

24 **MR. WEINGAERTNER:** That is correct. And also under
25 the same illustrative condition, not specifically taking --

1 **THE COURT:** Who invented the word "bytecode"? Where
2 did that come from? Is that in the patent?

3 **MR. WEINGAERTNER:** It comes up in the patent. It's
4 certainly -- my understanding is that it actually comes out of
5 the Java -- I'm sorry -- out of the virtual-machine concept
6 that predates Java. I think it goes back before that. I'm not
7 sure of the exact -- etymologically the first time it occurred
8 in the language, but my understanding of it is that the notion
9 of it, at least, goes back, you know, to the early days of
10 Java -- to the early days of virtual machines prior to Java.

11 **THE COURT:** Now, if you had -- if you were trying to
12 solve the problem of an application -- a single application
13 being written once, but run on anything, and you had a Mac and
14 a PC over there, so the intermediate bytecode would be
15 different; would wind up being different, I assume -- tell me
16 if I'm right -- so that it would trigger the right machine code
17 for the two respective --

18 In other words, the bytecode would be different for
19 the --

20 **MR. WEINGAERTNER:** No, your Honor.

21 **THE COURT:** For the PC, it would be different than
22 the one from the Apple?

23 No? Not right? Okay.

24 **MR. WEINGAERTNER:** It's a great question.

25 You could almost imagine the virtual machine running

1 in between here, taking the bytecode, and converting it to the
2 machine code, because that's where that actual layer and
3 additional level of processing and time that's required
4 resides: In this layer.

5 **THE COURT:** Something is omitted there. So there's
6 another --

7 **MR. WEINGAERTNER:** Sorry, your Honor?

8 **THE COURT:** There's another step there that's
9 omitted?

10 **MR. WEINGAERTNER:** No, your Honor. This just shows
11 the different kind of languages.

12 In the earlier ones, we just said, "How do you get
13 from one to the another?"

14 Here, in order to get from bytecode, you actually
15 have to bridge this gap, and bridge this gap here; but this
16 right here is actually running on the actual machine, so the
17 bytecode itself is portable.

18 **THE COURT:** So the bytecode would look the same for
19 the Apple as the would for a PC?

20 **MR. WEINGAERTNER:** Yes, your Honor.

21 **THE COURT:** I'm taking up too much of your time.
22 You've got about four or five more minutes, and then --

23 **MR. WEINGAERTNER:** We're happy to, whenever we can --

24 **THE COURT:** Okay.

25 **MR. WEINGAERTNER:** Maybe I can find an example to

1 skip to. Let's skip to Slide 39. I should have worn my
2 glasses.

3 Okay. This is the quote that I mentioned earlier.

4 And, by the way, your Honor, I just wanted to mention
5 one point. There was a discussion about class files earlier,
6 and I think there may have been a misapprehension. The concept
7 of a class, although present in Java, actually grew out of
8 object-oriented programming, which precedes Java. The C++
9 program language, your Honor, is one of those. And so Java
10 takes advantage of this notion of what a class is.

11 And so earlier we mentioned that there is -- there
12 are some interesting quotes from the folks who actually
13 developed Java; James Gosling being the person who is known as
14 the progenitor, in some ways, of Java, pointing out that
15 networking and Java go hand in hand, because the network allows
16 you to move things around to different machines really easily.

17 And so Java, again, in a way, kind of rode that
18 existing wave.

19 Next slide, please.

20 We wanted to mention something that had to do with
21 something that Mr. Jacobs pointed out, which is that there's a
22 notion that's at issue here involving what it means to be
23 computer readable, computer-readable medium. I want to not
24 really say much about the claim-construction issue, but he
25 thought there were some technical issues we should present that

1 would inform the discussion two weeks from now.

2 And the point is that -- I think if we turn back one
3 slide --

4 As Mr. Gosling said, getting Java out there is all
5 about making code available over network. It could be one's
6 handheld device. It could be a computer, like the one sitting
7 right in front of us right here. And the code is received over
8 a network onto a machine.

9 It's important, because the way some of the claims
10 are drafted specifically set out to kind of capture this notion
11 that the code is being transmitted.

12 Next slide. Thanks a lot.

13 And it could be transmitted from a local storage
14 device over a wire. It could be transmitted from a server over
15 the Internet. There could be any number of media that get
16 used, known primarily to big companies that have made huge
17 investments in telecommunications transmission equipment. And
18 it could be over a local area network; but in all instances,
19 it's being transmitted.

20 Next slide.

21 And these are just some examples of how data and code
22 ends up on our computers. Coaxial cable is well known. You
23 know, cable TV comes over coaxial cable. Copper wires, we all
24 know. Fiber is reasonably common. Infrared. People sometimes
25 will send handheld devices using infrared transmission.

1 Sometimes it's acoustic waves. Fax machines, for example, use
2 that. Light, which is, of course, related to fiber optics.
3 The point is there's a what's called a "carrier wave." This is
4 one representation that has a particular frequency. And signal
5 that you want to carry in it. This wave a modulated to sort of
6 impose this signal on this carrier. This is just a very
7 generic example, but that's how data actually winds up on
8 machines over the cables and over the Internet.

9 **THE COURT:** Is this an issue in the case?

10 **MR. WEINGAERTNER:** It is, your Honor. The definition
11 of what it means to be computer readable is an issue. So we
12 wanted to provide a little bit of a background for your Honor
13 to refer to at your Honor's wish; but that's really all we have
14 to say about that.

15 **THE COURT:** All right. What would you -- use the
16 rest of your time to take just one major issue, and lay out
17 what your view is and the opposing view is, so I can kind of
18 see where the two of you part company; but just on one major
19 issue that divides you here.

20 **MR. WEINGAERTNER:** Okay. One major issue. I guess
21 we could use this '104 Patent. It's the earliest-filed patent
22 of the group. Mr. Gosling's an inventor on this one. It's
23 drafted in some ways very broadly.

24 Interestingly, the patent was reissued, I think, more
25 than once, years after it originally filed. And the claims got

1 broader and broader. So we're dealing with the ones that tend
2 to be quite broad, and try to go down at the root level, to
3 what it means to resolve a data reference. We have a
4 difference in view as to what that means. It's one a few
5 examples. They're not -- they're sort of different issues from
6 one another, but I can certainly focus on this one.

7 And what I want to do is, if we could skip ahead a
8 little bit to Slide -- let's see here -- oh, jeez. Slide 45.

9 And Mr. Jacobs had very nice slides that showed, with
10 animation, how things worked. The idea here is that you're
11 resolving symbols down to numeric references, which actually
12 means taking something that is unspecified as to where it will
13 sit on a computer to a specific location. It's tied to a
14 specific place that you can go and find data.

15 And our understanding of the prior art, which is
16 actually referred to in the patent -- and we have lots of
17 others that go back also decades -- that this idea of having
18 to -- you know, to have the flexibility of a virtual machine,
19 but the speed of compilation, if you have to go and basically
20 resolve these each time you refer to them, it's time consuming.
21 And so the idea is to do it once, and not to do it again.

22 And the way that we believe the patent describes it
23 is by rewriting it, so that you needn't do it that over and
24 over again.

25 And the two parties have a difference in view as to

1 whether or not resolving actually requires rewriting, or
2 something less than that. So that's kind of a simply put issue
3 for one of the patents in this case, your Honor.

4 **THE COURT:** All right. Let's leave it there.

5 Mr. Jacobs, I'll give you two minutes if you feel
6 like there's some rebuttal you want to make, but I apologize
7 for not having more time. I need to bring this to a close.

8 **MR. JACOBS:** Your Honor, I think in many ways the
9 tutorials complemented each other; but there is one place in
10 Google's presentation there's a complete disconnect between
11 what we see, and what Google sees. And that's on this question
12 of the freedom of device manufacturers to change the code at
13 will.

14 Let me step back by saying that, in the Java world,
15 there is a whole regime set up to ensure that the interface to
16 the Java Virtual Machine, whoever implements it -- if it's
17 going to be called Java, it's going to meet it interface
18 definition in order to protect "Write once; run anywhere." If
19 the JVM across different machines is different, then the
20 bytecode won't execute properly.

21 Mr. Weingaertner pointed out that Java bytecode will
22 not execute on Android machines. From the Oracle side of the
23 world, that represents a fork in Java, because instead of
24 protecting "Write once; run anywhere," they've diverged from
25 it, and created a different interface. It has to be dex code

1 if you're in the Android world; not bytecode.

2 But within the Android world, they, too, want to
3 protect "Write once; run anywhere" as between diverse Android
4 platforms. So Google has told the users of Android -- the
5 device manufacturers -- to, wherever possible, as illustrated
6 in this slide, use the code available via the Android
7 open-source project -- excuse me -- rather than reimplement
8 significant parts of the system. Develop device implementers
9 are strongly encouraged to based their implementations, to the
10 greatest extent possible, on the upstream source code.

11 And, more recently, in the news, there have been
12 reports that Google is imposing very tight restrictions on its
13 licensees' ability to adjust the code residing on their
14 devices.

15 So, from a technical standpoint, we see both
16 companies, within their own worlds, aiming at very similar
17 things.

18 We aim, in the Java world, at Java compatibility
19 between diverse devices.

20 In the Android world, they aim at Android
21 compatibility at the Dalvik layer.

22 And so they have implemented a compatibility test
23 suite that has an analogy to the Java compatibility test suite.
24 And they have told their device implementers, "Use the Android
25 code that we make available."

1 **THE COURT:** We have to leave it there.

2 **MR. JACOBS:** Thank you very much, your Honor.

3 **THE COURT:** You all did a great job. And I look
4 forward to seeing you in two weeks. All right. Thank you.

5 (At 2:57 p.m. the proceedings were adjourned)

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

CERTIFICATE OF REPORTER

I, LYDIA ZINN, Official Reporter for the United States Court, Northern District of California, hereby certify that the foregoing proceedings in C. 10-3561 WHA, Oracle America, Inc., v. Google, Inc., were reported by me, a certified shorthand reporter, and were thereafter transcribed under my direction into typewriting; that the foregoing is a full, complete and true record of said proceedings as bound by me at the time of filing.

The validity of the reporter's certification of said transcript may be void upon disassembly and/or removal from the court file.

/s/ Lydia Zinn, CSR 9223, RPR

Friday, April 8, 2011